Swarm Intelligence and Flocking Behavior

Himani
M Tech Student
SLIET, Longowal

Ashish Girdhar
Assistant Professor
SLIET, Longowal

ABSTRACT
Swarm behavior suggests simple methodologies used by agents of swarm to solve complex problems, which using the other optimsation algorithms such as Genetic Algorithms may not be possible to solve. The basic reason behind this is the group behavior in these algorithms. The distributed control mechanism and simple interactive rules can manage the swarm efficiently and effectively. Flocking behavior does not involve central coordination. This paper aims at the review of the Swarm Intelligence algorithms developed so far and its association with flocking model.

General Terms
Artificial Intelligence, Swarm Intelligence, Algorithms

Keywords
Swarm Intelligence, agents, ACO, ABC, Flocking, PSO.

1. INTRODUCTION
The concept of SI was originally introduced by Gerardo Beni and Jing Wang[1] in 1989 in the context of cellular robotic systems (Beni and Wang, 1993), while Bonabeau et al[2] redefined it in 1999 as "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insect colonies and other animal societies". Most of the SI approaches are inspired by the collective behavior of natural species, such as ants foraging, birds flocking and bees gathering honey. Nevertheless, the term swarm could be extended to any constrained collection of interacting agents or individuals. This paper is organized as follows: Sections 2,3 and 4 lay down the basic principles which come under SI and associated algorithms. In section 5 we discuss the flocking behavior in detail and how it is different and/or similar to the swarm behavior and section 5.1 is discussion on PSO which is a mix of both swarm and flock. Finally in section 6 we conclude the paper.

2. SWARM INTELLIGENCE
Swarm Intelligence has been derived from the natural swarm behavior of animals which can be defined as the collective behavior exhibited by the animas of same size, aggregating together to solve a problem which is essential for their survival. SI can be defined as the emergent collective intelligence of groups of simple agents. Agents are analogous to the animals of the natural swarm. Agents can be a hardware device or a software program which operate in distributed control mechanism. Agents solve problems by interacting with other agents, or with their environment. Software agents are capable of taking simple decisions to solve a problem. SI has three fundamental and essential properties, namely decentralization, self-organisation and collective behaviour, which are necessary and sufficient to acquire SI behaviour. Millonas[3] proposed five principles to which the SI algorithms must adhere to. First is the proximity principle: the population should be able to carry out simple space and time computations. Second is the quality principle: the population should be able to respond to quality factors in the environment. Third is the principle of diverse response: the population should not commit its activities along excessively narrow channels. Fourth is the principle of stability: the population should not change its mode of behaviour every time the environment changes. Fifth is the principle of adaptability: the population must be able to change behaviour mode when it's worth the computational price. It is noticeable that the last two principles determine the performance of the SI algorithms. Table 1 lists some of the SI algorithms proposed so far.

Table 1: Entities and SI Algorithms associated

Entities	SI Algorithm
Ant	Ant Colony Optimisation[4]
Particles	Particle Swarm Optimisation[6]
Bees	Artificial Bee Colony[5]
Fireflies	Firefly Algorithm[7]
Monkeys	Monkey Search[8]
Cockroaches	Roach Infestation Optimisation[9]
Frogs	Jumping Frogs Optimisation[10]

3. ANT COLONY OPTIMISATION
The natural behaviour of ants during the foraging procedure inspires the optimisation algorithm as proposed by Colorni et al[4]. During the searching process for food sources, ants behave intelligently to find the optimal path to food source, which is practically achieved by the utilization of pheromone. The existence of pheromone shows the trace of an ant, and provides heuristic information for other ants, which decide whether to follow this pheromone trace or not. If the new ant chooses to follow this pheromone trace, it would reinforce the density of pheromone; otherwise the pheromone would be gradually evaporated and finally exhausted. The above decision strategies can be regarded as positive feedback and negative feedback respectively. Higher pheromone density indicates higher chosen probability. Therefore, more and more ants choose to follow the trace with high pheromone density and construct the optimal path to the food source.

The whole procedure of the ACO algorithm can be illustrated as follows. Initially, ants are placed on the nodes of the graph randomly. Then each ant selects a connected and unvisited node as its next movement probabilistically. The probability is influenced by two factors, the distance from the current node to the next node and the pheromone on the associated edge. This movement is executed iteratively until all ants have traversed all the nodes on the graph, which is called one cycle. After each cycle, the pheromone deployment of the whole graph is updated. The principle is that whenever an ant moves through an edge, the pheromone on that edge is reinforced; otherwise, it would evaporate and be exhausted. After a certain number of cycles, the path with highest pheromone

density is found, which represents the optimal solution. It may take some iterations or cycles to find the optimum path to the destination but the optimum path is eventually exploited. Thus the algorithm helps find the optimal solution without actually knowing the problem.

4. ARTIFICIAL BEE COLONY

This approach was proposed by Karaboga[5] on the principles of foraging by bees in the natural bee hives. When foraging, different bees work collaboratively to explore and exploit food sources with rich nectar. As per the proposed approach there are three types of bees: employed bees, onlookers and scouts. For every food source or candidate solution, there is an employed bee. The food sources with more nectar are better solutions to the problem. So the onlooker bees tend to eploit

The whole procedure of the ABC algorithm can be described as follows. Scout bees are assigned to find the initial food sources by carrying out a random search in the search space. After that, employed bees are sent out to exploit the discovered food sources, and each employed bee matches one food source. During the exploitation procedure, each employed bee also carries out a neighbourhood search and tries to find a better food source nearby. If a better food source is found, the employed bee would abandon the previous food source and exploit the better one. . After the completion of all employed bees, they return to the hive and share their information on food sources with onlooker bees waiting in the hive through a waggle dance. The onlooker bees would choose to follow certain employed bees and exploit corresponding food sources probabilistically. This probability is computed by the richness of the corresponding food sources. Once an onlooker bee chooses to follow an employed bee, it becomes an employed bee and repeats the procedure of employed bees. After certain number of iteration of the exploration and exploitation procedure, a food source may be exhausted and abandoned. In that case, the corresponding employed bee becomes a scout bee and randomly finds a new food source to replace the abandoned one. The whole procedure can divided into four phases: (1)initialization phase,(2)employed bee phase,(3)onlooker bee phase and(4)scout bee phase which is explained in the following figure.

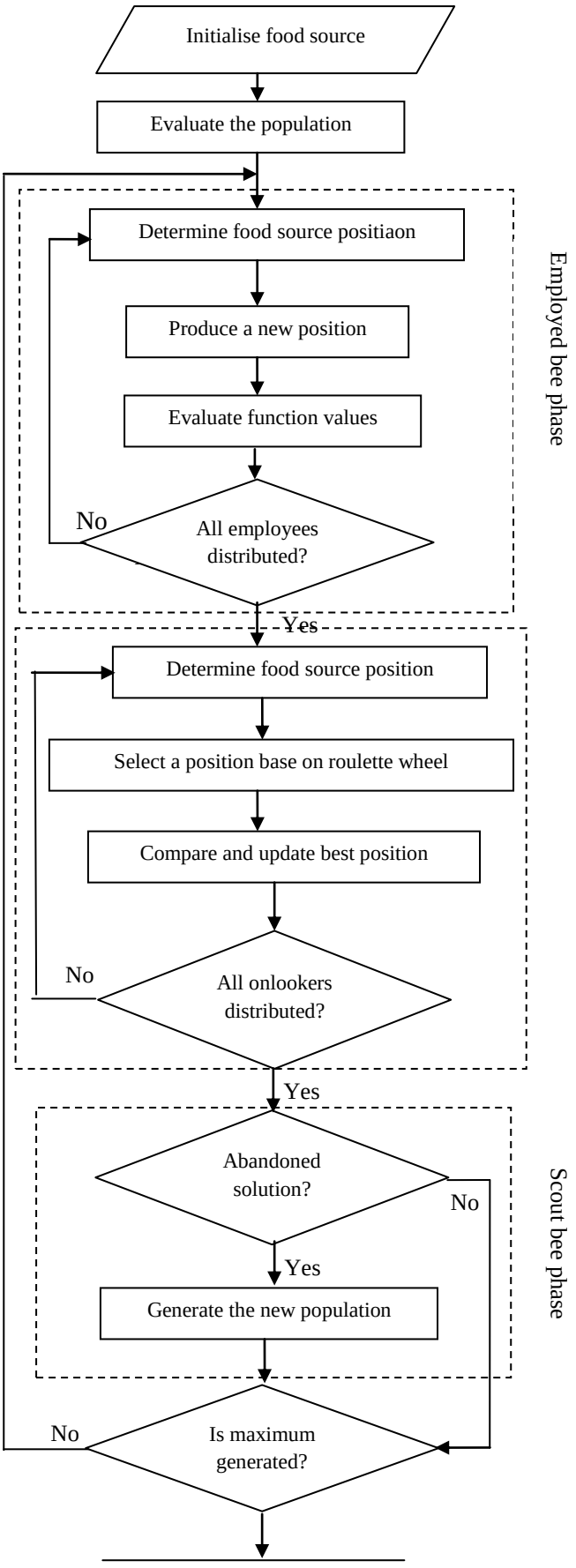

Fig 1: Flowchart of three phases of ABC

5. FLOCKING BEHAVIOUR

Flocking behaviour can be described as the behavior exhibited when a group of birds, called a flock, are foraging or in flight.[11] Birds and fish adjust their physical movement to avoid predators, seek food and mates, optimize environmental parameters such as temperature, etc. Birds have poor eyesight and they move in flocks in order to identify the obstacles in their paths. Similar algorithms can be applied in controlling the air traffic in the time of disaster management. Some fish such as clown fish also move in such flocking fashion called school. They use it to identify if some fish has been missed out in the journey. Humans, however adjust not only physical movement but cognitive or experiential variables as well. We do not usually walk in step and turn in unison rather, we tend to adjust our beliefs and attitudes to conform with those of our social peers**.** Essentially the basic models of flocking behaviour are controlled by three simple rules:

1. Separation - avoid crowding neighbours (short range repulsion)
2. Alignment - steer towards average heading of neighbours
3. Cohesion - steer towards average position of neighbours (long range attraction)

With these three simple rules, the flock moves in an extremely realistic way, creating complex motion and interaction that would be extremely hard to create otherwise.

The basic model has been extended in several different ways since it was proposed. In flocking simulations, there is no central control; each bird behaves autonomously. In other words, each bird has to decide for itself which flocks to consider as its environment. Usually environment is defined as a circle (2D) or sphere (3D) with a certain radius.

5.1 Particle Swarm Optimisation

It was proposed by James Kennedy and Russell Eberhart[6] in 1995 and was inspired by the bird flocking, fish schooling, and swarming theory. Combining the way birds and fish move in a flock and human social behaviour, this algorithm was proposed. They used the term particles in place of points, just to incorporate the idea of velocities associated with each particle. Therein each individual followed three simple rules: collision avoidance, velocity matching and flock centring. In addition to three rules, each particle has three inner attributes: it's historical best position (local best position) and global best position of the swarm, and refers to those two positions whenever it moves to the next position. The velocity of the particle is steered stochastically towards either of these two positions. The advantage of using an optimization method such as PSO is that it does not use the gradient of the problem to be optimized, so the method can be readily employed for a host of optimization problems. This is especially useful when finding the gradient is a cumbersome task or even impossible to derive. This versatility comes at a price, however, as PSO does not always work well and may need tuning of its behavioural parameters so as to perform well on the problem at hand.

The procedure of the algorithm can be explained as follows. At the beginning, a number of particles are randomly placed in the search space. Each particle holds its position and velocity information in a vector format. Whenever movement occurs, the particle needs to update its velocity information firstly, referring to three factors: its current velocity, the local best position and the global best position. Different weightings of different factors indicate different optimization strategies. Subsequently, the particle updates its position information following the updated velocity vector. The positions of each particle correspond to candidate solutions. The local and global best positions are updated after each movement provided that the particle arrives at a better position. This procedure is conducted iteratively until the stopping criteria are met. The global best position is the optimal solution which can be found so far.

for each particle i in [1 .. N]

initialize xi , vi

Pi = xi

End for

G = arg min f(Pi)

Pi

repeat:

for each Particle i in [1 .. N]

update vi

Check the velocity boundaries.

update xi

if f(xi) ≤ f(Pi) then Pi=xi

if f(Pi) < f(G) then G=Pi

end for

until stopping criterion is met

Fig 2:Pseudo-code for PSO

The basic elements in PSO are position and velocity. In this context, each solution is represented as one particle. Each particle has a position or in other words their fitness in the solution space. Each is then attached with a velocity which the particle will move towards the direction of the velocity vector pointing with the distance it specified. The particles will continue moving and updating the velocity until they reach a common goal or the optimum solution as all the particles agreed. Hence PSO reaches the global maximum value and is not affected by the problem of local maxima.

In addition to the population size, i. e. the number of candidate solutions, topology also affects the performance of the algorithm. However, practitioners tend to keep these constant across problems, although there is some evidence to suggest that bigger populations are better for higher dimensional problems and that highly-connected topologies work better for unimodal(single maxima/minima point) problems, while more sparsely-connected topologies are superior on multimodal problems. Dynamic problems are challenging for PSO. These problems are modelled by the fitness functions which change over time and so storing the values in the memory is obsolete. So, to date a fully comprehensive mathematical model of particle swarm optimization is still not available. There are several reasons for this. Firstly, the PSO is made up of a large number of interacting elements (the particles). Although the nature of the elements and of the interactions is simple, understanding the dynamics of the whole is nontrivial. Secondly, the particles are provided with memory and the ability to decide when to update the memory. This means that from one iteration to the next, a particle may be attracted towards a new p_i(personal best velocity of particle i) or a new p_g(global best velocity of

all the particles) or both. Thirdly, forces are stochastic. This prevents the use of standard mathematical tools used in the analysis of dynamical systems. Fourthly, the behaviour of the PSO depends crucially on the structure of the fitness function. However, there are infinitely many fitness functions, and so it is extremely difficult to derive useful results that apply to all, even when complete information about the fitness function is available.

6. CONCLUSION

The nature has provided us with a technique for processing information that is at once elegant and versatile.SI and flocking behaviour are based on the swarming methodologies already present in nature. ACO uses natural ants' foraging system to find the optimal path to the destination. ABC uses bees' way of building swarm and exploiting the nectar to search for the optimum solution. Based on the swarming theory and flocking of birds, the PSO algorithm is apparently simple to implement. Some parameters and fitness function however has to be readjusted for applying it to different problems. Nonetheless, in the last few years, a number of theoretical advances have been made. Social optimization occurs in the time frame of ordinary experience - in fact, it is ordinary experience. It can be concluded that this class of algorithms belong ideologically to that philosophical school that allows wisdom to emerge rather than trying to impose it, that emulates nature rather than trying to control it, and that seeks to make things simpler rather than more complex.

7. REFERENCES

[1] Beni, G., Wang, J., 1993. Swarm Intelligence in Cellular Robotic Systems, Robots and Biological Systems: Towards a New Bionics Springer, Berlin Heidelberg.

[2] Bonabeau,E.,Dorigo,M.,Theraulaz,G.,1999.Swarm intelligence: from natural to artificial systems.

[3] Millonas, M. M. (1994). Swarms, phase transitions, and collective intelligence. In C. G. Langton, Ed., Artificial Life III**.** Addison Wesley.

[4] Colorni, A.,Dorigo, M.,Maniezzo,V.,1991.Distributed optimization by ant colonies. In: Proceedings of the first European Conference on Artificial Life.

[5] Karaboga, D., 2005. An idea based on honey bee swarm for numerical optimization.

[6] Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory (MHS'95). In: Proceedings of the Sixth IEEE International Symposium on Micro Machine and Human Science.

[7] Yang,X.-S.,Deb,S.,2009.Cuckoo search via Lévy flights. (NaBIC2009).In: IEEE World Congress on Nature and Biologically Inspired Computing.

[8] Mucherino, A., Seref, O., 2007. Monkey Search: A Novel Metaheuristic Search for Global Optimization, Data Mining, Systems Analysis and Optimization in Biomedicine. American Institute of Physics, New York.

[9] Havens, T.C., Spain, C.J., Salmon, N.G., Keller, J.M., 2008. Roach infestation optimization. In: Proceedings of the IEEE Swarm Intelligence Symposium.

[10] Garcia, F.J.M., Perez, J.A.M., 2008. Jumping frogs optimization: a new swarm method for discrete optimization.

[11] http://en.wikipedia.org/wiki/Flocking_(behavior)

[12] Walter J. Gutjahr,2006. Mathematical runtime analysis of ACO algorithms: survey on an emerging issue

[13] Ali Kaveh, Advances in Metaheuristic Algorithms for Optimal Design of Structures.

[14] Shuzhu Zhang, C.K.M.Lee, 2014. Swarm intelligence applied in green logistics: A literature review.